Distributed Throughput Optimization over P2P Mesh Networks using Diffusion Adaptation

Zaid J. Towfic, Jianshu Chen, and Ali H. Sayed

Department of Electrical Engineering
University of California, Los Angeles
Email: {ztowfic, jshchen, sayed}@ee.ucla.edu

***Abstract*—This work develops a decentralized adaptive strategy for throughput maximization over peer-to-peer (P2P) networks. The adaptive strategy can cope with changing network topologies, is robust to network disruptions, and does not rely on central processors. The algorithm is obtained as a special case of a more general diffusion strategy for the distributed solution of optimization problems with constraints. Simulation results illustrate how the proposed technique is competitive with other methods.**

***Index Terms*—diffusion adaptation, P2P networks, mesh networks, throughput maximization, distributed processing, barrier function.**

I. Introduction

Peer-to-peer networks are useful for file sharing and for distributing bandwidth in streaming video applications [1], [2]. One application is video-on-demand (VoD) systems where users request video streams from a server. While legacy implementations involve a client requesting video directly from a server, recent implementations allow the server to stream a few nodes directly and these nodes will in turn stream the popular video streams to other nodes.

In early P2P video streaming systems, the server creates and maintains a tree network [1]–[3]. Tree overlays do not efficiently utilize bandwidth resources and are less robust to network disruptions since a failure at an "older" parent node will cause failure at all nodes beneath it. For these reasons, mesh networks have been advanced as an attractive alternative to tree networks [4], [5]. In mesh networks, a single client can have multiple sources and can also serve as a parent node to multiple child nodes.

One problem that requires a distributed solution over such P2P mesh networks is the problem of determining the rates that can be accommodated over the links. As was already well illustrated in [5], such problems and similar network utility maximization problems [6], give rise to linear programs (LPs) that can be solved centrally at the server location. This approach, however, requires the server to know the constraints of all nodes in the system. In addition, when nodes enter or leave the system, the server would need to re-solve the updated LP and communicate the new rates to all nodes in the network.

This work was supported in part by NSF grants CCF-1011918, and CCF-0942936.

In [5], the centralized optimization problem was transformed into a decoupled optimization problem by introducing its dual formulation [6]. The problem was then solved using sub-gradient iterations for the dual variables with diminishing step-sizes. At every iteration, each node communicates its dual variables to its neighbors, and receives its neighbors' dual variables. This exchange allows each node to estimate what the outgoing rates should be on its links. One drawback of the method is the communication overhead associated with sharing of the dual variables. In addition, when nodes enter or leave the system, a new round of optimization needs to be initiated by the server since the algorithm utilizes a diminishing step-size.

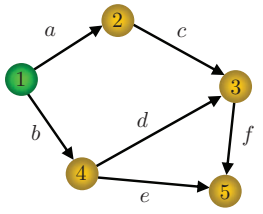
In this paper, and motivated by recent work on adaptation over networks [7]–[10], we propose an adaptive diffusion strategy for the direct optimization of the link-rates in the network. Unlike [5], we solve the primal problem directly as opposed to the dual problem. We extend the general adaptive diffusion strategies from [7]–[10] to optimization problems that involve a set of inequality constraints for each node. This step is achieved by aggregating the inequality constraints into a barrier function and defining appropriate localized cost functions at each node. The approach allows us to solve the network throughput optimization problem by only communicating the link-rates themselves (or measuring the link-rates at the receiver nodes). In addition, the adaptive diffusion strategy utilizes a *constant* step-size and, as such, is capable of tracking the link-rates even when the network topology changes slowly. The ability to track changes in network topology is a highly desirable feature given the dynamic nature of P2P mesh networks.

II. Problem Formulation

A. Optimization Problem

In this section, we review the centralized optimization problem formulated in [5]. Consider a connected network consisting of N nodes and L links in a directional graph. We denote the set of all the links as $\mathcal{L}$. We let node 1 denote the server. The network is created by nodes accessing the server and the server connecting new nodes to already existing nodes in the network. We describe the network topology using two

978-1-4577-2053-6/12/$31.00 ©2012 IEEE



$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Fig. 1. Sample streaming network. The shaded node is the server.

matrices A and B to represent the outgoing and incoming edges, respectively. The elements of A and B are defined as:

$$a_{k,l} = \begin{cases} 1, & \text{if link } l \text{ leaves node } k \\ 0, & \text{otherwise} \end{cases}$$

$$b_{k,l} = \begin{cases} 1, & \text{if link } l \text{ arrives at node } k \\ 0, & \text{otherwise} \end{cases}$$

We denote the rows of the $\mathbb{R}^{N \times L}$ matrices A and B by a_k^T and b_k^T:

$$A = \operatorname{col}\left\{a_1^T, a_2^T, \ldots, a_N^T\right\}, \quad B = \operatorname{col}\left\{b_1^T, b_2^T, \ldots, b_N^T\right\}$$

For illustration purposes, consider the simple connected network of Fig. 1 with 5 nodes and 6 edges. The corresponding matrices A and B are shown in the figure. Our objective is to maximize the total throughput through the network where we define throughput in the same manner as in [5]. Throughput is defined as the total received rate excluding packets that arrive after the needed time (after the segment of video has already been played). Specifically, we seek to minimize the following cost function (subject to certain constraints discussed further ahead):

$$-\sum_{k=1}^{N}\left[\pi_k \sum_{l \in \mathcal{L}} b_{k,l} w_l p_l - \sum_{l \in \mathcal{L}} \epsilon a_{k,l} w_l^2\right] \tag{1}$$

where p_l is the expected probability a packet arrives by the playback deadline over link l, π_k denotes the positive weight given to node k's throughput, and w_l is the rate associated with edge l. Different nodes can have different weights π_k since their throughput depends on the type of service they have purchased. The small positive scalar ϵ is a regularization factor that makes the objective function strongly-convex in the variables w_l. We form a vector $w \triangleq \operatorname{col}\{w_1, \ldots, w_L\}$ that collects the optimization variables for all edges in the graph.

Each node has an upload bandwidth constraint imposed by the Internet service provider (ISP). The constraint can be expressed mathematically for each node k as

$$Aw \preceq u \tag{2}$$

where the output constraints are in the column vector u and the notation $\preceq$ represents component-wise comparisons. In addition, the download bandwidth is also capped by a user's ISP or by a user's network administrator so that the user does not excessively consume the local network resources. We write this constraint as

$$Bw \preceq \min\{d, s_r\} \tag{3}$$

where we have limited the download rate by either a constraint d_k or the source rate, s_r, whichever is smaller.

We must note that no node can have an output link with a rate higher than the total incoming rate into that node. For the server, this constraint is modified so that no output link from the server can have a rate higher than the source rate s_r. The constraint is written as

$$a_{k,l} w_l \leq a_{k,l}\left[\sum_{m \in \mathcal{L}} b_{k,m} w_m + s_r \delta_{k-1}\right], \begin{array}{l} k \in \{1, \ldots, N\} \\ l \in \mathcal{L} \end{array} \tag{4}$$

The symbol δ_n represents the Kronecker delta function that is equal to 1 when $n = 0$ and is zero otherwise. In our case, we have represented the server as node 1, and the server has no incoming nodes; thus the constraint becomes $w_l \leq s_r$ for all outgoing edges from the server. Finally, a non-negativity constraint is placed on the rate variable so that

$$-w \preceq 0 \tag{5}$$

We can now form the optimization problem by combining the objective function (1) with the constraints (2)-(5) to get the following convex program, which is equivalent to the one proposed in [5]:

$$\begin{aligned} \min_{w} \; & -\sum_{k=1}^{N}\left[\pi_k\left(\sum_{l \in \mathcal{L}} b_{k,l} w_l p_l\right) - \sum_{l \in \mathcal{L}} \epsilon a_{k,l} w_l^2\right] \\ \text{subject to} \; & \\ & a_k^T w \leq u_k, \quad \forall k \\ & b_k^T w \leq \min\{d_k, s_r\}, \quad \forall k \\ & a_{k,l} w_l \leq a_{k,l}\left[\sum_{m \in \mathcal{L}} b_{k,m} w_m + s_r \delta_{k-1}\right], \; \forall k, l \\ & -a_{k,l} w_l \leq 0, \quad \forall k, l \end{aligned} \tag{6}$$

B. Using Barrier Functions

Reference [5] proceeds to solve (6) by introducing the dual problem and by using sub-gradient iterations with diminishing step-sizes. At this point, we diverge and solve (6) by instead relying on adaptive diffusion strategies that work directly with the primal problem. Doing so will lead to enhanced performance and tracking, as the simulations further ahead illustrate.

We first convert (6) into an unconstrained optimization problem by employing barrier functions (as in interior-point methods [11]). Barrier functions are functions that attain nearly zero values when a constraint is satisfied while quickly changing to a high value when the constraint is violated.

Rather than present the adaptive diffusion solution by working with the special case (6), we shall instead motivate the approach by considering a general optimization problem with constraints. Afterwards, we return to (6) and specialize the algorithm to the case of P2P networks. Thus, consider an optimization problem of the following general form:

$$\begin{aligned} \min_{w} \quad & \sum_{k=1}^{N} f_k(w) \\ \text{subject to} \quad & g_{k,m}(w) \leq 0, \quad k \in \{1,\ldots,N\} \\ & \qquad\qquad\quad m \in \{1,\ldots,M_k\} \end{aligned} \tag{7}$$

where the inequality constraints are split on a per-node basis. The functions $\{f_k(\omega), g_{k,m}(w)\}$ are all assumed to be convex. In (7), each node k has a set of M_k inequality constraints $\{g_{k,1},\ldots,g_{k,M_k}\}$. We embed the constraints into the objective function by using a barrier function $\phi(x)$:

$$\min_{w} \quad \sum_{k=1}^{N} \left[f_k(w) + \sum_{m=1}^{M_k} \phi\left(g_{k,m}(w)\right) \right] \tag{8}$$

The objective function that appears in (8) is the sum of N local cost functions, one for each node, defined as:

$$J_k(w) \triangleq f_k(w) + \sum_{m=1}^{M_k} \phi\left(g_{k,m}(w)\right) \tag{9}$$

so that (8) becomes:

$$\min_{w} \quad J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w) \tag{10}$$

There are several useful choices for the barrier function $\phi(\cdot)$; see Sec. III-C further ahead.

Our objective is to solve (10) in a distributed manner. This is necessary because solving (10) at the server requires the latter to know the capacity constraints across all nodes in the network (and nodes join and leave the network at random times). In addition, the nodes themselves are not aware of the full network structure, of the connection matrices A and B, and may only have access to a single row of each of these matrices. Following arguments similar to [7]–[10], and especially [12], an adaptive diffusion algorithm can be derived to solve (10). Each node k would run the following recursions:

$$\begin{cases} \psi_{k,i} = w_{k,i-1} - \mu_k \nabla_w J_k(w_{k,i-1}) \\ w_{k,i} = \sum\limits_{m \in \mathcal{N}_k} c_{m,k} \psi_{m,i} \end{cases} \tag{11}$$

where $\mathcal{N}_k$ denotes the neighborhood of the node k (including itself) and the matrix C formed out of the elements $c_{m,k}$ is a doubly-stochastic combination matrix.

III. Throughput Optimization Via Diffusion Adaptation

Motivated by algorithm (11), in this section we derive a distributed adaptive algorithm for the P2P throughput optimization problem (6).

A. Local Costs

We define the local cost functions $J_k(w)$ as:

$$\begin{aligned} J_k(w) = & -\pi_k b_k^T (w \odot p) + \epsilon a_k^T (w \odot w) + \\ & \phi(a_k^T w - u_k) + \phi(b_k^T w - \min\{d_k, s_r\}) + \\ & \phi\left(a_k \odot (w - \left((b_k^T w) + s_r \delta_{k-1}\right) \mathbb{1}\right) + \\ & \phi\left(-a_k \odot w\right) \end{aligned} \tag{12}$$

where the notation $a \odot b$ indicates the Hadamard (entry-wise) product of two vectors a and b. In the above, we overloaded the notation so that when the barrier function is a function of a vector in $\mathbb{R}^L$, then the scalar barrier function operates on each element independently. Compared to [12], we notice an interesting aspect of the local costs in (12), in that each one of them depends only on a subset of the unknown vector w. The corresponding gradient vector is given by:

$$\begin{aligned} \nabla_w J_k(w) = & -\pi_k p \odot b_k + 2\epsilon (w \odot a_k) - \\ & \nabla\phi\left(-a_k \odot w\right) \odot a_k + \nabla\phi(a_k^T w - u_k) a_k + \\ & \nabla\phi(b_k^T w - \min\{d_k, s_r\}) b_k + \\ & \nabla\phi\left(a_k \odot (w - \left((b_k^T w) + s_r \delta_{k-1}\right) \mathbb{1}\right) \odot a_k \end{aligned}$$

B. Adaptive Diffusion Algorithm

We observe that $\nabla_w J_k(w)$ is non-zero only for the links that node k has direct access to (those are the links that either arrive to, or leave from, the node). We shall refer to the set of links that node k is directly connected to as $\mathcal{L}_k$:

$$\mathcal{L}_k \triangleq \{l : a_{k,l} + b_{k,l} > 0\} \tag{13}$$

The set of all link-rate estimates at time i that are associated with this set is denoted by $w_{k,\mathcal{L}_k,i}$, with $w_{k,l,i}$ denoting the particular estimate for link $l \in \mathcal{L}_k$. Since each node does not have access to the size of the full vector w or the topology of the network, it is not necessary for each node k to estimate the full vector w but only $w_{k,\mathcal{L}_k}$. Thus, at every iteration, each node k runs Alg. 1 in order to optimize the link-rates.

Algorithm 1 Adaptive Diffusion Throughput Optimization

Each node starts with some estimate $w_{k,l,-1}$ of rates w_l for links $l \in \mathcal{L}_k$. The matrix $C \in \mathbb{R}^{N \times L}$ is formed from nonnegative entries $c_{m,l}$ and is left-stochastic with $c_{m,l} = 0$ if $l \notin \mathcal{L}_m$. For each time instant $i \geq 0$, each node k repeats:

for all $l \in \mathcal{L}_k$ **do**

$\psi_{k,l,i} \leftarrow w_{k,l,i-1} - \mu_k \nabla_{w_{k,l}} J_k(w_{k,\mathcal{L}_k,i-1})$

end for

for all $l \in \mathcal{L}_k$ **do**

$w_{k,l,i} \leftarrow \sum\limits_{m=1}^{N} c_{m,l} \psi_{m,l,i}$

end for

where $w_{k,\mathcal{L}_k,i-1}$ indicates all available previous link-rate estimates (of links connected to node k).

It should be noted that Alg. 1 does not require explicit communication of the estimates $\psi_{m,l,i}$ from a transmitter node to a receiver node since this information can be measured at

the receiver node. Measurements of the link rates can have some errors associated with them and such errors can be addressed in a manner similar to [13]. Actually, since these nodes are direct neighbors in the P2P network, the estimates $\psi_{m,l,i}$ can also be transmitted (rather than measured) since the communication overhead to transmitting the link-rates is negligible when compared to communicating the actual stream. We assume some protocol is formulated to govern this interaction. Such transmissions can occur with acknowledgements as set out in the RTSP protocol [14].

C. Choice of Barrier Function

As stated earlier, there are many possible choices for the barrier function $\phi(x)$. One choice that is commonly used in practice is the log-barrier function [11] $\phi_{\rm lb}(x) = -\frac{1}{t}\log(-x)$. The log-barrier function approximates the indicator function:

$$I(x) = \begin{cases} \infty, & x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

as $t \to \infty$. It is desirable to choose a function that attains finite values and avoids numerical overflow and has a continuous derivative. For this reason, we use the barrier function:

$$\phi(x) = \begin{cases} t\sqrt{(x+\tau)^2 + \rho^2}, & x + \tau > 0 \\ t\rho, & \text{otherwise} \end{cases} \tag{15}$$

for some small positive constants ρ and τ and large positive constant t. The constant τ allows us to shift the barrier slightly inside the feasible set in order to reduce the infeasibilities.

IV. Simulation Results

In this section, we compare the algorithm to the centralized solution of the regularized problem found via CVX [15] and to the distributed algorithm proposed in [5]. We generate a network of $N = 200$ nodes consisting of $3N$ links where the server can have at most $0.1N$ neighbors and all other nodes are limited to at most 7 neighbors. The network is guaranteed to be connected. The centralized problem utilizes a regularization factor $\epsilon = 0.01$ in order to ensure that the solution to the regularized problem is close to the solution of the original LP.

The constraints are chosen to be similar to [5] so that:

- Server upload capacity is 1 Gbps.
- The probability p_l is uniform in the range $(0.75, 1]$.
- The network has two types of users:
  - Type 1: 15% of the network has upload capacity uniform in the range $[0.5, 1)$ Mbps and download capacity uniform in the range $[1, 2)$ Mbps.
  - Type 2: 85% of the network has upload capacity uniform in the range $[1, 2)$ Mbps and download capacity uniform in the range $[10, 20)$ Mbps.
- All users have the same weight $\pi_k = 1$.

We choose $t = 1$ as this avoids overflow in the algorithm while still attaining more feasible solutions than [5], as we will show through the simulations. We select values of $\rho = 0.001$ and $\tau = 0.02$ and use a combination weight of $1/2$ for each of the two nodes to combine the estimates of the two nodes. For Alg. 1, we choose a constant step-size of $\mu_k = 0.01$ for all nodes. Both our algorithm and the method of [5] start from a trivially feasible rate vector of $w = 0$. The time is computed assuming data are observed (or communicated) at the average network throughput rate where packets are assumed to be 1024 bytes long. As a performance metric, we use the relative error:

$$\boxed{P = \frac{T_{\text{alg},i}}{T_{\rm CVX}}} \tag{16}$$

where $T_{\text{alg},i}$ indicates the average network throughput for the algorithm at time i and $T_{\rm CVX}$ indicates the average network throughput obtained from the centralized solution solved by CVX. We also plot the maximum value of the constraints in the original problem. This is a good indicator of how feasible the solutions obtained by the algorithms are. For example, if an algorithm finds a vector w that yields a near-optimal solution to the original problem in terms of throughput, but has less feasible entries, then this solution should be less desirable as a similarly performing vector in terms of the network throughput value, with more feasible entries. Among the advantages of the proposed method are that it enables better tracking when the network topology changes and it provides better performance when the nodes do not know the probabilities p_l but have to rely on estimates for them.

A. Tracking Performance

We first conduct an experiment to evaluate the tracking performance of the proposed algorithm. In this experiment, it is seen that diminishing step-sizes prevent proper tracking. To illustrate this effect, we select for [5] a diminishing step-size of the form $\mu_{k,i} = \omega/i$ for the algorithm in [5] where $\omega = 1.3$ since this step-size decays quickly and allows us to simulate the tracking performance once the step-size has decayed sufficiently. In this simulation, 50 nodes join the network around the 115-second mark and each node k is assumed to know the probabilities p_l for the links in $\mathcal{L}_k$ exactly. The new nodes are added to the network according to the same properties discussed in the previous subsection. For example, the capacities are still selected according to the two user-type structure presented in Sec. IV. Fig. 2 shows the tracking performance of Alg. 1. We notice that as the network changes, Alg. 1 quickly reaches the feasibility floor dictated by t and τ by adjusting the estimated rates. The algorithm proposed in [5] would require some trigger to all nodes in order to reset their diminishing step-sizes. This point can be observed in practice when the step-size sequence falls below machine precision and would need to be reset in order for the algorithm to continue the adjustment process. The adaptive diffusion algorithm, on the other hand, continues to operate and is able to track changes in network topology in real-time.

B. Fixed-Network Link Rate Learning

Let's now illustrate the performance of the algorithm when the network is fixed. Figure 3 shows the performance of the two algorithms when each node k knows p_l exactly for $l \in \mathcal{L}_k$.

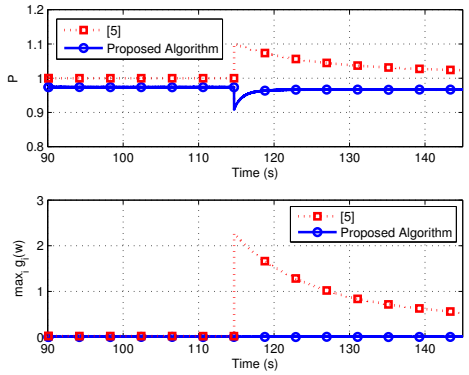


Fig. 2. Tracking performance of link-rate optimization algorithms. The top plot shows the normalized average throughput over the network. Values close to one indicate better performance. The bottom plot shows the adherence of the algorithms to the constraints. Lower values indicate better performance.

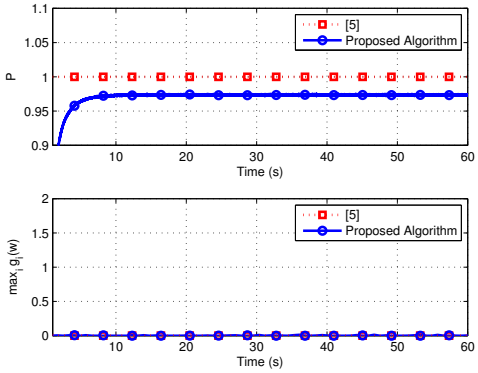


Fig. 3. Performance of the proposed algorithm and [5]. The top plot shows the normalized average throughput over the network. Values close to one indicate better performance. The bottom plot shows the adherence of the algorithms to the constraints. Lower values indicate better performance.

We choose a diminishing step-size of the form $\mu_{k,i} = \omega/\sqrt{i}$ used in [5] with $\omega = 1.3$. We must note that improving the performance in terms of feasibility involves increasing t or decreasing τ and in turn decreasing μ_k. We can see that the algorithm proposed in [5] has an advantage in this scenario due to its precise convergence when p_l's are known.

However, generally, the nodes do not know the true probabilities p_l but only observe variables x_l that are Bernoulli distributed with success probability p_l. Therefore, the expected value of the *observed* cost function yields $J_k(w)$ in (12). This case is presented in Fig. 4. After approximately 60 seconds, the proposed algorithm achieves a network throughput that is approximately 2.71% below that of the optimal solution of the regularized centralized optimization problem. The throughput generated by the algorithm proposed in [5] exhibits fluctuations around the optimal solution. We also know that the solution from [5] can violate some constraints, as shown in the bottom plot in Fig. 4—this is the reason why the throughput can be above that of the centralized solution at times.

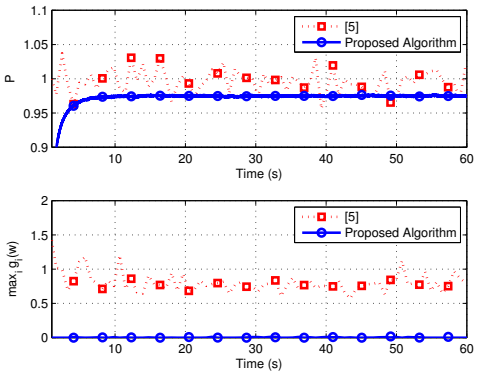


Fig. 4. Performance of the proposed algorithm and [5] for a fixed network, where the nodes do not know the probabilities p_l but instead observe Bernoulli variables with success rate p_l. The top plot shows the normalized average throughput over the network. Values close to one indicate better performance. The bottom plot shows the adherence of the algorithms to the constraints. Lower values indicate better performance.

V. Conclusion

In conclusion, we presented a method for solving strongly-convex optimization problems where the constraint set is distributed over a network of nodes. Our approach does not require a central processor and operates on interactions between network neighbors. We showed that our approach can track changes in the network without a reset command from a central entity and cope with unknown parameters.

References

[1] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. ACM NOSSDAV*, Miami Beach, Florida, May 2002, pp. 177–186.

[2] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," Technical Report 2001-30, Stanford InfoLab, April 2001.

[3] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems," in *Proc. ACM SIGMETRICS*, Annapolis, Maryland, June 2008, pp. 169–180.

[4] T. Hossain, Y. Cui, and Y. Xue, "Rate distortion optimization for mesh-based P2P video streaming," in *Proc. IEEE ICC*, Dresden, Germany, June 2009.

[5] Y. He, I. Lee, and L. Guan, "Distributed throughput maximization in P2P VoD applications," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 509 –522, April 2009.

[6] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal of Selected Topics in Communication*, vol. 24, no. 8, pp. 1439–1451, August 2006.

[7] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7-2, pp. 3122–3136, June 2008.

[8] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1035–1048, March 2010.

[9] Z. J. Towfic, J. Chen, and A. H. Sayed, "Collaborative learning of mixture models using diffusion adaptation," in *Proc. IEEE MLSP*, Beijing, China, September 2011, pp. 1–6.

[10] J. Chen and A. H. Sayed, "Performance of diffusion adaptation for collaborative optimization," in *Proc. IEEE ICASSP*, Kyoto, Japan, Mar. 2012.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[12] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *Submitted for publication. Available as Arxiv preprint arXiv:1111.0034*, Oct. 2011.

[13] S-Y. Tu and A. H. Sayed, "Adaptive networks with noisy links," in *Proc. IEEE GLOBECOM*, Houston, Texas, December 2011.

[14] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," RFC 2326, IETF, Apr. 1998.

[15] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, April 2011.